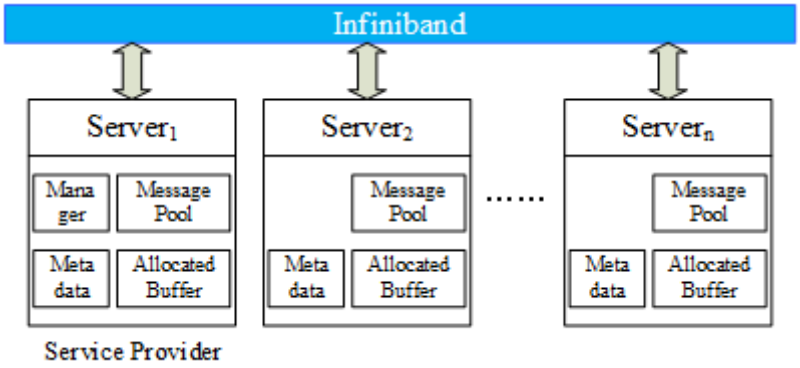
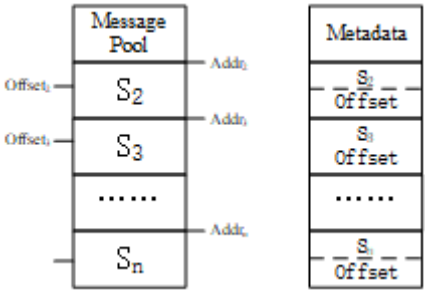
Uniform Memory Design

笔记本:	内存计算		
创建时间:	2018/11/19 15:32	**更新时间:**	2018/11/22 19:04
作者:	simba_wei		

所谓统一内存，而不是共享内存这一概念，是指打破了节点之间内存的物理隔离，实现了跨越不同节点的内存寻址和内存管理。利用现有的RDMA网络，我们可以实现不同节点上内存数据间接近总线的访问速率。基于这一设定，我们可以将不同机器节点上的内存看做一整片的内存空间（类似单机环境中，一台机器的DDR卡槽上插着多根内存条）。如下图所示，节点1、2、……、n中的Allocated buffer组成了一片连续的内存空间。



每个节点上都有一个消息处理池，这个消息处理池是一个片固定的memory buffer。在server启动的时候，内存根据固定的配置文件切分成固定的区间用于存储不同节点传输过来的消息。如下图所示，作为节点1上的消息处理池，它负责存储2到n节点的消息。与此同时，节点1的metadata中也记录了他发送的消息在其他节点的消息池中的位置和偏移量，以便于能够快速直接的往目标节点中写入自己所需要的信息。



每台机器启动时候会读取配置文件，根据配置中的信息来确定这台机器上所需分配的内存大小，并汇报给管理节点。

```
    void Free(char*);
private:
    char* memBlock_;  //the big block of memory that will be added into unified shared memory
```

```
MemoryManager::MemoryManager(size_t size)
{
    // 1. allocate a big block of memory
    memBlock_ = new char[size];
```

而对Manager函数而言，它实际上所要管理的就是这好几片 membuffer (char类型)。这几片buffer首先需要注册到RDMA设备当中，然后它们所申请的内存的大小以及起始地址都需要进一步的汇报给Manager模块运行的节点上。在Manager模块中统一实现malloc （暂时不允许跨越机器的内存块的分配）和 free 的具体函数功能。

其中Malloc函数如下图所示。函数参数中给定具体数值的大小，返回值为char*指针，指向刚刚分配的内存。内存释放函数Free需要再参数中给出对应内存块的头指针，函数会在内存管理的链表中标记为释放，并加到free的内存双向链表中。

```
/**
 * new interface for allocate memory
 * @param size of allocated memory
 * @return the pointer of allocated memory
 */
char* Malloc(size_t);
```

```
/**
 * release the memory that has been allocated
 * @param the pointer of allocated memory that should be released
 */
void Free(char*);
```

内存的管理节点会根据机器中内存使用的情况，将内存分配的指令发送到对应机器上，由指定的机器去负责具体内存的分配任务。内存分配得到的block的信息加上节点本身的信息组成该快内存block的全局地址，为接下来的全局共享的数据存储提供信息基础。